

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-227403

(43) 公開日 平成8年(1996)9月3日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/16	3 7 0		G 0 6 F 15/16	3 7 0 N
9/46	3 4 0		9/46	3 4 0 B
12/02	5 4 0		12/02	5 4 0
13/00	3 5 5	7368-5E	13/00	3 5 5
// G 0 6 F 9/40	3 1 0		9/40	3 1 0 C

審査請求 未請求 請求項の数10 F D (全 14 頁)

(21) 出願番号 特願平7-207405

(22) 出願日 平成7年(1995)7月24日

(31) 優先権主張番号 2 7 9 0 9 3

(32) 優先日 1994年7月22日

(33) 優先権主張国 米国 (U S)

(71) 出願人 591064003

サン・マイクロシステムズ・インコーポレ
ーテッド

SUN MICROSYSTEMS, IN
CORPORATED

アメリカ合衆国 94043 カリフォルニア
州・マウンテンビュー・ガルシア アヴェ
ニュー・2550

(72) 発明者 パナギオティス・エス・クーギオリス

アメリカ合衆国 94040 カリフォルニア
州・マウンテンビュー・デイル アヴェニ
ュ・ナンバー132・1200

(74) 代理人 弁理士 山川 政樹

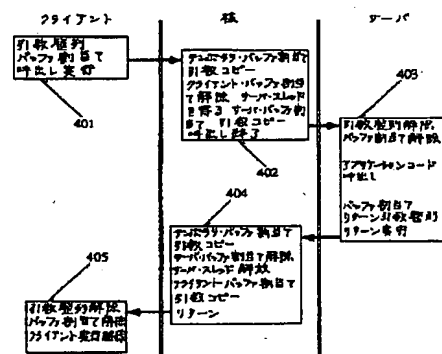
最終頁に続く

(54) 【発明の名称】 スペース効率に優れたプロセス間通信方法及び装置

(57) 【要約】

【課題】 スペース効率の高いプロセス間通信方法及び装置を提供する。

【構成】 第1のプロシージャが、この第1のプロシージャと第2のプロシージャによって共用される第1のメモリスペース中に第1のバッファを割当てて、第1のプロシージャは、第2のプロシージャと通信するための引数を第1のバッファ中に整列させる、第2のプロシージャのためのメッセージが現在渡されているということを指示するとともに、第1のメモリスペース中の第1のバッファに対する第1の参照を第2のプロシージャに渡す。第2のプロシージャは、第1のプロシージャによるメッセージの指示を検出した後、第1のバッファを参照し、第1のバッファ中の引数をテンポラリ・バッファにコピーする。次に、第2のプロシージャは、第1のバッファを割当て解除する。



【特許請求の範囲】

【請求項 1】 a. クライアント・プロシージャが、そのクライアント・プロシージャとカーネル・プロシージャによって共用される第 1 のメモリスペース中に第 1 のバッファを割当ててステップと、

b. 上記クライアント・プロシージャが、リモート・プロシージャの呼出しのための引数を上記第 1 のバッファに整列させるステップと、

c. 上記クライアント・プロシージャが、上記カーネル・プロシージャを介して上記リモート・プロシージャを呼出し、上記第 1 メモリスペースにおける上記第 1 のバッファに対する第 1 の参照を上記カーネル・プロシージャに渡すステップと、

d. 上記カーネル・プロシージャが、上記クライアント・プロシージャによる上記リモート・プロシージャの上記呼出しを検出し、そのカーネル・プロシージャと上記サーバ・プロシージャによって共用される第 2 のメモリスペース中に第 2 のバッファを割当ててステップと、

e. 上記カーネル・プロシージャが、上記第 1 のバッファを参照し、上記第 1 のバッファ中の上記引数を上記第 2 のバッファにコピーするステップと、

f. 上記カーネル・プロシージャが、上記第 1 のバッファを割当てて解除するステップと、

g. 上記カーネル・プロシージャが、上記リモート・プロシージャを呼出し、上記第 2 のバッファに対する第 2 の参照をそのリモート・プロシージャに渡すステップと、

h. 上記リモート・プロシージャが、そのリモート・プロシージャの上記呼出しを検出し、上記第 2 のバッファ中の上記引数を整列解除するステップと、

i. 上記リモート・プロシージャが上記第 2 のバッファを割当てて解除し、実行するステップと、を具備したコンピュータシステムにおけるコンピュータ実装型のプロセス間通信方法。

【請求項 2】 a. 上記リモート・プロシージャの実行終了と同時に、そのリモート・プロシージャが上記第 2 のメモリスペース中に第 3 のバッファを割当ててステップと、

b. 上記リモート・プロシージャが、そのリモート・プロシージャからリターンするために、上記第 3 のバッファにリターン引数を整列させるステップと、

c. 上記リモート・プロシージャが、上記カーネル・プロシージャにリターンし、そのカーネル・プロシージャに上記第 3 のバッファに対する第 3 の参照を渡すステップと、

d. 上記カーネル・プロシージャが、上記リモート・プロシージャの上記リターンを検出し、上記第 1 のメモリスペースの中に第 4 のバッファを割当ててステップと、

e. 上記カーネル・プロシージャが、上記第 3 のバッファ中の上記リターン引数を上記第 4 のバッファにコピー

するステップと、

f. 上記カーネル・プロシージャが、上記第 3 のバッファを割当てて解除するステップと、

g. 上記カーネル・プロシージャが、上記クライアント・プロシージャにリターンし、上記第 4 のバッファに対する参照を上記クライアント・プロシージャに渡すステップと、

h. 上記クライアント・プロシージャが、上記クライアント・プロシージャへのリターンを検出し、上記第 4 のバッファ中の上記引数を整列解除するステップと、

i. 上記クライアント・プロシージャが、上記第 4 のバッファを割当てて解除し、実行を続けるステップと、をさらに具備した請求項 1 記載の方法。

【請求項 3】 a. 第 1 のプロシージャが、その第 1 のプロシージャと第 2 のプロシージャによって共用される第 1 のメモリスペース中に第 1 のバッファを割当ててステップと、

b. 上記第 1 のプロシージャが、上記第 2 のプロシージャとの通信のための引数を上記第 1 のバッファに整列させるステップと、

c. 上記第 1 のプロシージャが、上記第 2 のプロシージャに関するメッセージを指示し、上記第 1 のメモリスペース中の上記第 1 のバッファに対する第 1 の参照をその第 2 のプロシージャに渡すステップと、

d. 上記第 2 のプロシージャが、上記第 1 のプロシージャによる上記メッセージの上記指示を検出するステップと、

e. 上記第 2 のプロシージャが、上記第 1 のバッファを参照し、その第 1 のバッファ中の上記引数をテンポラリ・バッファにコピーするステップと、

f. 上記第 2 のプロシージャが、上記第 1 のバッファを割当てて解除するステップと、を具備したコンピュータシステムにおけるコンピュータ実装型のプロセス間通信方法。

【請求項 4】 a. 上記第 2 のプロシージャが、上記テンポラリ・バッファ中の上記引数を処理するステップと、

b. 上記引数の処理の終了と同時に、上記第 2 のプロシージャが、上記第 1 のメモリスペース中に第 2 のバッファを割当ててステップと、

c. 上記第 2 のプロシージャが、その第 2 のプロシージャからリターンするために、リターン引数を上記第 2 のバッファに整列させるステップと、

d. 上記第 2 のプロシージャが、上記第 1 のプロシージャにリターンし、その第 1 のプロシージャに上記第 2 のバッファに対する第 2 の参照を渡すステップと、

e. 上記第 1 のプロシージャが、その第 1 のプロシージャへのリターンを検出し、上記第 2 のバッファ中の上記引数を整列解除するステップと、

f. 上記第 1 のプロシージャが、上記第 2 のバッファを

3

割当て解除し、実行を続けるステップと、をさらに具備した請求項3記載の方法。

【請求項5】 a. 第1のプロシージャと第2のプロシージャとによって共用される第1のメモリスぺース中に第1のバッファを割当てる第1の回路と、

b. 上記第2のプロシージャと通信するために上記第1のプロシージャによって参照された引数を上記第1のバッファに整列させる第2の回路と、

c. 上記第2のプロシージャに関するメッセージを指示し、上記第1のメモリスぺース中の上記第1のバッファに対する第1の参照を上記第1のプロシージャからその第2のプロシージャに渡す第3の回路と、

d. 上記第1のプロシージャによる上記メッセージの上記指示を検出する第4の回路と、

e. 上記第1のバッファを参照し、上記第1のバッファ中の上記引数を上記第2のプロシージャによって使用されるテンポラリ・バッファにコピーする第5の回路と、

f. 上記第1のバッファを割当て解除する第6の回路と、を具備したプロセス間通信装置。

【請求項6】 a. 上記テンポラリ・バッファ中の上記引数を処理する第7の回路と、

b. 上記引数の処理終了と同時に動作して、上記第1のメモリスぺース中に第2のバッファを割当てる第8の回路と、

c. 上記第2のプロシージャからリターンするために上記第2のバッファにリターン引数を整列させる第9の回路と、

d. 上記第1のプロシージャにリターンし、上記第2のバッファに対する第2の参照をその第1のプロシージャに渡す第10の回路と、

e. 上記第1のプロシージャへのリターンを検出し、上記第2のバッファ中の上記引数を整列解除する第11の回路と、

f. 上記第2のバッファを割当て解除し、実行を続ける第12の回路と、をさらに具備した請求項5記載の装置。

【請求項7】 a. 第1のプロシージャが、その第1のプロシージャと第2のプロシージャによって共用される第1のメモリスぺース中に第1のバッファを割当てるステップと、

b. 上記第1のプロシージャが、上記第2のプロシージャとの通信引数を上記第1のバッファに整列させるステップと、

c. 上記第1のプロシージャが、上記第2のプロシージャに関するメッセージを指示し、上記第1のメモリスぺース中の上記第1のバッファに対する第1の参照をその第2のプロシージャに渡すステップと、

d. 上記第2のプロシージャが、上記第1のプロシージャによる上記メッセージの上記指示を検出するステップと、

4

e. 上記第2のプロシージャが、上記第1のバッファを参照し、その第1のバッファ中の上記引数をテンポラリ・バッファにコピーするステップと、

f. 上記第2のプロシージャが、上記第1のバッファを割当て解除するステップと、を具備したプロセス間通信方法を実装したコンピュータシステム。

【請求項8】 a. 上記第2のプロシージャが、上記テンポラリ・バッファ中の上記引数を処理するステップと、

10 b. 上記引数の処理の終了と同時に、上記第2のプロシージャが、上記第1のメモリスぺース中に第2のバッファを割当てるステップと、

c. 上記第2のプロシージャが、その第2のプロシージャからリターンするために、リターン引数を上記第2のバッファに整列させるステップと、

d. 上記第2のプロシージャが、上記第1のプロシージャにリターンし、その第1のプロシージャに上記第2のバッファに対する第2の参照を渡すステップと、

e. 上記第1のプロシージャが、その第1のプロシージャへのリターンを検出し、上記第2のバッファ中の上記引数を整列解除するステップと、

f. 上記第1のプロシージャが、上記第2のバッファを割当て解除し、実行を続けるステップと、をさらに具備した請求項7記載の方法を実装したコンピュータシステム。

【請求項9】 a. 第1のプロシージャと第2のプロシージャによって共用された第1のメモリスぺースの中に第1のバッファを割当てる第1の割当て回路と、

b. 上記第1のプロシージャが、上記第2のプロシージャと通信する引数を上記第1のバッファに整列させることを可能にする第1の整列回路と、

c. 上記第1のプロシージャからのメッセージを上記第2のプロシージャに指示し、上記第1のメモリスぺース中の上記第1のバッファに対する第1の参照を上記第2のプロシージャへ渡す第1のメッセージ指示回路と、

d. 上記第2のプロシージャが上記第1のプロシージャによる上記メッセージの上記指示を検出することを可能にする第1のメッセージ検出回路と、

e. 上記第2のプロシージャが上記第1のバッファを参照し、上記第1のバッファ中の上記引数をテンポラリ・バッファにコピーすることを可能にする第1の参照回路と、

f. 上記第1のバッファ中の上記引数の上記テンポラリ・バッファへのコピーの終了と同時に動作して、上記第2のプロシージャが上記第1のバッファを割当て解除することを可能にする第1の割当て解除回路と、を具備したコンピュータシステム。

【請求項10】 a. 上記第2のプロシージャが上記テンポラリ・バッファ中の上記引数を処理することを可能にする処理回路と、

b. 上記引数の処理終了と同時に動作して、上記第2のプロシージャが上記第1のメモリスぺースの中に第2のバッファを割当ててを可能にする第2の割当て回路と、

c. 上記第2のプロシージャが、その第2のプロシージャからリターンするために、上記第2のバッファ中にリターン引数を整列させることを可能にする第2の整列回路と、

d. 上記第2のプロシージャが上記第1のプロシージャにリターンし、上記第2のバッファに対する第2の参照をその第1のプロシージャに渡すことを可能にするリターン回路と、

e. 上記第1のプロシージャがその第1のプロシージャへの上記リターンを検出し、上記第2のバッファ中の上記引数を整列解除することを可能にする第2の検出回路と、

f. 上記第1のプロシージャが上記第2のバッファを割当て解除し、実行を続けることを可能にする第2の割当て解除回路と、をさらに具備した請求項9記載のコンピュータシステム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、コンピュータシステムに関するものである。より詳しくは、本発明は、コンピュータシステムにおけるプロセスの間通信に関する。

【0002】

【従来の技術】プロセス間通信は、現代のコンピュータシステム設計の一つの基本的な部分である。プロセス間通信は、通常、クライアントとサーバ・プロセスとの間の通信を管理するコンピュータシステムのカーネルの呼出し技法によって容易に行える。このようなプロセス間通信に付随する問題の1つは、例えばクライアントタスクは、サーバタスクを呼び出すとき引数（パラメータ）をそのサーバタスクに渡すためにメモリの一定量を割当て、そのメモリは通常何らかの引数がサーバから戻されるまで使用状態に置かれるということである。すなわち、制御及び処理がサーバ・アプリケーションに渡された後も、サーバからリターン引数が戻されるまで、依然としてメモリがクライアント・アプリケーションで費消される。このように、サーバタスクがクライアントから制御を渡された時間の大半の間にわたって使用されることなく一つのバッファが割当てられている。これは、各スレッドにつき1つずつというように複数のバッファが割当てられるマルチスレッド環境においては、特に問題となる。すなわち、多数のバッファが割当て状態のままに置かれ、その大部分が各スレッドの持続時間の間未使用状態に保たれ、メモリ資源が不必要に費消される。

【0003】従来技術におけるプロセス間通信の典型的な仕組みを図1に示す。典型的には、クライアントタスク（例えば図1の110）は、通常、クライアント・プ

ロセス及びカーネル・プロセスに利用可能なバッファまたは他の保護メモリスぺースである符号111で示すようなある量のメモリスぺースを割当て、そのバッファ領域に引数を整列させる。本願において、「整列」とは、クライアント・プロセスがサーバ・プロセスに渡される引数、パラメータまたは他のデータがあるメモリエリアに格納するプロセスを指す。セキュリティないしは安全保護の理由から、このメモリエリアは、通常クライアント・プロセス110及びオペレーティング・システム100のカーネルのみが利用可能である。

【0004】図1に示すサーバ・プロセス120に対するコールが起こると、カーネル100がそのコールを検出し、制御はカーネルに渡される。この場合、クライアントは、通常ポインタまたは参照（reference）をメモリエリア（バッファ）111に渡し、その後、カーネルはバッファ111に渡された引数にアクセスすることができる。次に、カーネル100は、そのメモリエリア111に対する参照を受け取り、メモリエリア111内にある引数をカーネル100のテンポラリ・メモリエリアにコピーする。すると、カーネルがアクセス可能な第2のメモリエリア101及びサーバルーチン120を用いて、カーネル100からサーバ120へ通信することができる。テンポラリ・バッファ中の引数は、メモリエリア（バッファ）121にコピーされる。次に、プロセス120用のサーバルーチン・スレッドが生成され、カーネル100によってこのスレッドに対する参照が行われる。サーバ120は、カーネル100からのメモリエリア121に対する参照を受け入れ、かつ引数を整列解除する。

【0005】サーバ・プロセス120がアクティブのときは、カーネル100による呼出し後も、バッファ111はクライアント・プロセス110に割当てられた状態になっている。一部の従来技術のアプリケーションにおいては、引数を整列させるためのバッファは、長さが5キロバイトのオーダーである。クライアントは、このメモリエリアをサーバ・プロセス120の実行期間中、オープンでアクセス可能な状態に保つ。サーバ・プロセス120の実行終了と同時に、上記のクライアント／サーバ呼出しプロセスの逆のプロセスが行われ、その際、サーバは、それ自身のバッファ121を用いて引数をカーネル100のメモリエリアに整列させるとともに、参照がこのカーネル100のメモリエリアに渡される。最終的には、リターン引数は、クライアント・プロセス110中の元のバッファ領域111内に戻る。典型的な従来技術のシステムにおいては、この時になって初めて、クライアント・プロセス及びサーバ・プロセスのバッファ111及び121が両方とも割当て解除される。このように、これらのバッファは、カーネル100及びサーバ・プロセス120がアクティブであり、またおそらくはアイドルである（例えばI/O処理待ちの間）長い時間

7

にわたって、割当てられたまま、アイドル状態に保たれることになる。これは、メモリ資源の無駄遣いである。さらに、マルチスレッド環境においては、クライアント・プロセス110及びサーバ・プロセス120が111及び121のようなバッファを多数割当てて、対応する呼び出された側のプロセスによって制御がリターンされるのを待つ間、それらのバッファを割当て状態に保つ場合がある。これは、メモリ資源が大量かつ不必要に費消される結果につながる。

【0006】

【発明が解決しようとする課題】コンピュータシステムにおけるスレッド数が増大するにつれて、このような通信によって費消されるメモリの量が非常に重要な意味を持つようになる。従って、本発明は、プロセス間通信におけるメモリ資源の使用に付随する従来技術の欠点を解消するためになされたもので、その目的は、スペース効率の高いプロセス間通信方法及び装置を提供することにある。

【0007】

【課題を解決するための手段】本発明は、コンピュータシステムにおけるプロセス間通信用のコンピュータで実行される方法及び装置にある。本発明においては、第1のプロシージャが、この第1のプロシージャ（例えばクライアント・プロセス）と第2のプロシージャ（例えばカーネルまたはサーバ・プロセス）によって共用される第1のメモリスぺース中に第1のバッファを割当てる。次に、第1のプロシージャは、第2のプロシージャと通信するための引数を第1のバッファ中に整列させる。また、第1のプロシージャは、第2のプロシージャのためのメッセージが現在渡されているということを指示するとともに、第1のメモリスぺース中の第1のバッファに対する第1の参照を第2のプロシージャに渡す。第2のプロシージャは、第1のプロシージャによるメッセージの指示を検出する。次に、第2のプロシージャは、第1のバッファを参照し、第1のバッファ中の引数をテンポラリ・バッファにコピーする。すると、第2のプロシージャは、第1のバッファを割当て解除することができる。本発明の実施の形態においては、第1のバッファは第2のプロシージャによる通信の受取りと同時に割当て解除されるので、プロセス間通信がより効率的になる。

【0008】次に、第2のプロシージャは、テンポラリ・バッファ中の引数を処理することができる。引数を処理し終わったならば、第2のプロシージャは、リターン引数を整列させるために、第1のメモリスぺース中で第2のバッファを割当てる。第2のプロシージャは、第1のプロシージャにリターンし、これに第2のバッファに対する第2の参照を渡す。第1のプロシージャは、この第1のプロシージャへのリターンを検出し、第2のバッファ中の引数を整列解除する。第1のプロシージャは、第2のバッファを割当て解除し、実行を継続する。本発

8

明の実施の形態においては、第1のメモリスぺースが第1の数のバッファを参照する第1のプロシージャによる第2のプロシージャの呼出しに先立って、第1のメモリスぺースが第1のサイズに事前割当てされる。上記第1の数の各バッファを第1または第2のプロシージャによって割当て、それらのバッファが第1または第2のプロシージャによって使用中であることを指示することも可能である。上記第1の数の各バッファが第1のプロシージャまたは第2のプロシージャによって使用中であることの指示は、割当てフラグとバッファが使用中であることを示す値との間のアトミック・スワップによって行われ、その際上記割当てフラグは上記第1の数のバッファの1つが使用中であるかどうかを示すために用いられる。

【0009】以下、本発明を添付図面に示す実施の形態により詳細に説明する。これらの実施の形態は、例示説明を目的とするものであり、本発明に対して制限的な意味を有するものではない。なお、添付図面中、同じ参照符号は同じ構成要素を示す。

【0010】

【実施の形態】本発明は、プロセス間通信、特にリモート・プロシージャ呼出しを実装したコンピュータシステムにおけるプロセス間通信のより効率的な方法を提供するものである。本発明は、特定の実施の形態との関連、特に汎用プログラム・コンピュータシステムとの関連において説明するが、当業者には、本発明がその要旨及び範囲を逸脱することなく様々なシステムで実施すること可能なことは明白であろう。本発明は、一連のデータ構造及びコンピュータシステム中で動作可能なコンピュータ・プログラムに実装される付随命令として実施される。このようなデータ構造は、図2のブロック図に示すようなコンピュータシステムで生成することができる。

【0011】図2において、符号200は、本発明の一実施の形態が実装されるコンピュータシステムを示す。コンピュータシステム200は、情報を伝達するためのバスまたはその他の通信手段201、バス201と結合された情報を処理するための処理手段202を具備する。システム200は、さらに、情報及びプロセッサ202によって実行される命令を記憶するためのバス201に結合されたランダムアクセスメモリ（RAM）またはその他の書換可能な記憶装置204（以下主メモリと称する）を具備する。また、主メモリ204は、プロセッサ202による命令実行の間、一時的変数またはその他の中間情報を記憶するためにも使用することができる。さらに、システム200は、プロセッサ202用の静的情報及び命令を記憶するためのバス201と結合されたリードオンリーメモリ（ROM）及び／またはその他の静的記憶装置206、及び磁気ディスクまたは光ディスクとこれらに対応するディスクドライブのようなデータ記憶装置207も具備している。データ記憶装置2

07は、情報及び命令を記憶するために、バス201と結合されている。この記憶装置は、商業ベースで入手可能なソフトウェア製品を用いて、その時現在定義済みの問題記述についての情報を維持する以下に説明するデータベースの記憶に使用することも可能である。

【0012】コンピュータシステム200は、情報をコンピュータのユーザに表示するために、バス201に接続されたブラウン管(CRT)や液晶表示装置(LCD)のような表示装置221を具備することもできる。さらに、このような表示装置221は、表示装置221上に表示される単一または複数のフレームまたは画像のような情報を記憶するフレームバッファ210を介してバス201に接続することもできる。バス201には、情報及びコマンド・セレクションをプロセッサ202に伝達するための文字・数字キー及びその他のキーを含む文字・数字入力装置222を接続することも可能である。他のユーザ入力装置として、バス201には、方向情報及びコマンド・セレクションをプロセッサ202に伝達するとともに、表示装置221上のカーソルの動きを制御するためのマウス、トラックボール、スタイラス、あるいはカーソル方向キーのようなカーソル・コントロール223が接続されている。

【0013】ここで、コンピュータシステム200のコンポーネント及び関連ハードウェアは、その一部または全部を様々な態様で使用することができるが、各特定の態様により、種々の目的に応じて、システムのどのような構成でも使用することが可能であるということは理解できよう。

【0014】一実施の形態においては、システム200は、米国カリフォルニア州マウンテンビューのSun Microsystems(登録商標)社の製造になるSPARCstationブランドのワークステーションのようなSun Microsystems(登録商標)のブランド・ファミリーのワークステーションの1つである。プロセッサ202としては、上記Sun Microsystems(登録商標)社の製造になるSPARCブランドのマイクロプロセッサの一種を用いることができる。

【0015】以下の様々な実施の形態の説明においては、高水準プログラミング言語(例えば、CまたはC++言語)で生成され、例えば、上記の米国カリフォルニア州マウンテンビューのサン・マイクロシステムズ社より入手可能なSPARC compilerによって実行時にシステム200でコンパイル、リンクされた後、目的コードとして実行される一連のルーチンに具体的に言及する。詳しく言うと、本発明は、米国カリフォルニア州マウンテンビューのSunSoft社から入手可能なSolaris(登録商標)のスレッド・パッケージのような一部のソフトウェアライブラリと共に使用することができる(Sun、Sun Microsystems

及びSolarisは、カリフォルニア州マウンテンビューのSun Microsystems社の登録商標である。SPARC及びSPARCstationは、スパーク・インターナショナル(SPARC International, Inc.)社の登録商標であり、サン・マイクロシステムズ(Sun Microsystems)社に専用実施権が許諾されている。しかしながら、以下に説明する方法及び装置は、離散型の論理デバイス、大規模集積回路(LSI)、特定用途向け集積回路(ASIC)、またはその他の専用ハードウェアのような特殊目的のハードウェア装置に実装することも可能であるということは理解できよう。本願の説明は、本願と同様の機能を有する装置にも等しく適用可能である。

【0016】次に、本発明の実施の形態について図3以降の図面を参照しつつ説明する。本発明の一実施の形態においては、プロセス間通信は2つのプロセス間の共有バッファスペースによって行われる。これらのプロセスは、1つの例においては、クライアント・プロセスとカーネルであり、他の場合においては、カーネルとサーバ・プロセスである。このようにして、クライアントとサーバ・プロセスの間の呼出しの時の引数の渡し操作のような通信を、以下に説明するメカニズムを用いて行うことができる。クライアント・プロセス、カーネル及びサーバを有するシステムにおいては、クライアントとサーバの間の通信を行うために2つのバッファ領域が使用される。これについては、図3及び4により図式的に説明する。図3に示すように、クライアント・プロセス310は、クライアント・プロセス310とカーネル・プロセス(ルーチン)300の間の通信に使用する複数のバッファ(各々約5キロバイト)を割当てる。同様に、カーネル・プロセス300とサーバ・プロセス320は、同様のサイズの複数のバッファ321を介して通信する。クライアント/カーネル間及びカーネル/サーバ間の通信には、「必要に応じて」、さらにバッファが割当てられる。このように、第1のプロセスでバッファを割当て、そのバッファの割当て状態を呼出しの間(通常呼出しからのリターンまで)維持する従来技術と異なり、本発明の実施の形態においては、受け取り側のプロセスに呼出し側のプロセスから情報を受け取り次第、バッファを割当て解除させる。この動作を図4に図式化して示す。

【0017】例えば、図4に示すように、第1のプロセス(例えば、図3の310)は、外部プロセスの呼出しと同時に引数を整列させ、バッファを割当てた後、その外部プロセスの呼出しを実行する。これは、図4のステップ401に示されている。場合によっては、外部プロセスは、「リモート」と呼ばれる。すなわち、この場合、外部プロセスは、第1のプロセス(クライアント)のアドレス空間へのアクセスを持たず、逆に第1のプロ

セスも外部プロセスのアドレス空間へのアクセスがない。同様に、分散型環境においては、このリモート・プロセスは、ローカル・プロセッサ、またはコンピュータシステムのメモリ、あるいはリモートのコンピュータシステムのリモート・プロセッサに常駐するものであってもよい。図3に示すカーネル・プロセス300（別名「核」）による呼出しの検出と同時に、クライアントにより渡される引数を受け取るためのテンポラリー・バッファ（例えば図3の301）が割当てられる。クライアント・バッファからカーネル内のテンポラリー・バッファに引数がコピーされると同時に、クライアント・バッファは割当て解除される。このように、本実施の形態においては呼出し期間の間ずっとクライアント・バッファの割当て状態を維持するのではなく、クライアント・バッファは、カーネルが適切な引数を受け取り、コピーし終えるまでしか使用されない。

【0018】カーネルに適切な引数がコピーされたならば、カーネルは引数を渡す第2のプロセスのためのサーバ・スレッドを得、カーネルとサーバの間の通信用に適切なバッファ（例えば図3の321）が割当てられる。引数がテンポラリー・バッファからサーバ・バッファへのコピーされた後、制御はサーバ・プロセス320に渡される。このステップは、図4の402に示す。サーバ・プロセス320による呼出しの検出と同時に、サーバは、サーバ・バッファ321から引数を受け取って整列解除する。この時、サーバは、そのサーバ／カーネル・バッファ321を割当て解除し、図4のステップ403に示すように実行することができる。

【0019】サーバ・プロセスの実行終了と同時に、サーバ・バッファは割当てられ、クライアント・プロセス300はリターンするための引数を整列させることができる。次に、リターン引数は、サーバ／カーネル・バッファ321に整列させられ、サーバ・プロセス320からのリターンが行われる。カーネル300によるリターンの検出と同時に、カーネル内でテンポラリー・バッファが再度割当てられ、そこにリターン引数がコピーされ、その後、サーバ・バッファ321は割当て解除される。すると、カーネルは、サーバ・スレッドを解放し、カーネル300とクライアント310の間の通信のためにクライアント・バッファを割当てることができる。この点で、カーネルは、ステップ404で引数を逆にクライアント／カーネル・バッファ311にコピーし、制御のクライアント310へのリターンが行われる。ステップ405におけるカーネルからクライアントへの制御のリターンが検出されると、クライアントは、バッファ中にある引数を整列解除し、バッファが割当て解除される。次に、クライアントは、サーバ・プロセス320からリターン引数がリターンされた後、実行を継続することができる。バッファ321は、再び次のスレッドを実行可能な状態になる。

【0020】このように、本発明においては、クライアントとサーバの間の通信を行うためのバッファが、「必要に応じて」動的に割当てられる。バッファは、2つのプロセス間の通信にとって必要でなくなると、「割当て済みバッファ」フラグのクリアによって割当て解除される。これは、第2のプロセス（例えばサーバ）からのリターンを待つバッファを割当て解除する従来技術と対照的な特徴である。現代のコンピュータシステムに共通に用いられているようなマルチスレッド環境においては、各呼出しスレッド毎の別個のバッファの割当てのため、及び呼出しの間これらのバッファを割当て状態に維持するために多量のメモリ資源が費消される。通常、このような従来の技術的情況においては、サーバ・プロセスへの呼出し期間の間、これらのバッファは使用されない状態に置かれる。このように、本発明は、第2のプロセス（例えば、サーバ・プロセス320）の実行の間に必要とされない多数のスレッドにバッファを割当てることによる大量のメモリ使用を回避することによって、プロセス間通信のためのより効率的な手段を提供するものである。従って、本発明の実施の形態においては、そのような従来技術におけるプロセス間通信よりはるかに効率的にメモリが使用される。

【0021】本発明の実施の形態では、このプロセス間通信を容易にするための図5に示すようなデータ構造を使用する。図5において、符号500は制御域を示す。これは、例えば、クライアント／カーネル制御域（あるいはカーネル／サーバまたは他のその他の通信領域）であってもよい。この領域は、2つのプロセスの間に必要な通信を行うために用いられる。制御域500は、どちらか通信を出す側のプロセスによってアクセスされるポインタを介して、複数の事前割当てされたバッファ513～519を参照する。これらは、プロセス間通信時にクライアント、カーネルまたはサーバ・プロセスによって実際にデータが格納されるバッファ領域である。制御域500は、現在割当てられているバッファ数を表す整数値を入れる第1のフィールド501を有する。図5に示す例においては、フィールド501は整数値4が書き込まれており、現在4つのバッファ領域（513、515、517及び519）が割当てられていることを示している。本発明の実施の形態においては、フィールド501は、最大値として32を書き込む（最大32のバッファを参照する）ことができるが、これは単に設計上の選択に関する問題であり、使用するバッファの数は32より多くても少なくてもよい。

【0022】また、制御域500は、所与のバッファ領域が現在割当てられているかどうかを示す割当てフラグを有する。図に示すように、フィールド502、504、506、508は、次のフィールド（ポインタまたは参照）が現在あるスレッドに割当てられているバッファまたはメモリエリアを指示しているかどうかを示す

「alloc/dealloc」フラグを有する。フィールド503、505、507、509等は、バッファ自身へのポインタまたは参照であり、各々前のフィールドのalloc/deallocフラグと対応する。例えば、フィールド502中のフラグの値は、フィールド503中のポインタによって参照されたバッファ513が現在割当てられているか、割当て解除されているかを指示する。バッファ513～519は、各々小さいメモリエリアからなり、例えば、本発明の実施の形態においては、5キロバイトである。これらのバッファ領域は、フィールド501を調べて制御域によって参照された中で利用可能なバッファがあるかどうかを検知し、かつ各alloc/deallocフラグ（例えば、502、504等）を調べて、その調べている特定のバッファが使用可能であるかどうかを決定することにより、クライアント、カーネルまたはサーバ・プロセスによって「必要に応じて」割当てられる。

【0023】クライアント・プロセス（例えば図3の310）の初期化と同時に、第1のプロセス（例えばクライアント）から第2のプロセス（例えばカーネル300）への何らかの通信時における使用のために制御域500が割当てられる。現在使用中のバッファ数が生成中の全スレッドにとって不十分なときは、各々のバッファに使用される実メモリスペースも、「必要に応じて」オペレーティング・システムから割当てられる。図8に示すようなもう一つの実施の形態においては、nの全てのバッファ（n=32）に対してメモリを一度に割当てることが可能である（例えばクライアント・プロセスに入ると同時に、またはスレッドの最初の生成と同時に）。

【0024】第2のプロシージャの呼出しが検出されると、以下に図6により説明するようにしてバッファを割当てることができる。この処理は、クライアント、カーネルまたはサーバが通信しているプロセスへ、あるいはそのプロセスから引数を渡すためにバッファを割当てるとき行うことができる。本発明の一実施の形態においては、バッファは、第2のプロセスの呼出しの間に常使用可能である。本発明のもう一つの実施の形態においては、クライアントは、共用バッファのいずれかを使用する前に、最小量のメモリ（例えば128バイト）以上のメモリ量が必要かどうかを決定することも可能である。

【0025】上記のいずれの場合にも、プロセス600は、ステップ602で開始され、まずカウンタを図5の要素501のようなアレイA[0]中の第1の要素に等しくセットする。インデックスは1に初期化される。次に、ステップ604で、カウンタが0から許容最大バッファ数である32までの指定範囲外であるかどうか決定される。この範囲外であれば、ステップ606で、割当てプロセスからエラーがリターンされる。そうでなければ、ステップ608で、カウンタが正確に0に等しいかどうか判断される。カウンタの内容0は、生成しよ

うとするスレッドに現在利用可能な使用されていないバッファがないということを示し、プロセスはステップ610にリターンして、今回割当てに利用可能な空のバッファはないということを指示する。この場合、プロセスは、異常終了するか、バッファが利用可能になるまで待つか、あるいはバッファを含む第2のメモリエリアを割当てることができる。

【0026】さらにプロセス600の説明を続けると、ステップ604でカウンタの内容が上記範囲外でないと判断されるか、ステップ608で正確に0に等しくないことが検出された場合は、ステップ612で仮の値が0に等しくセットされる。上に説明した実施の形態で使用する規定においては、フィールド502、504等の1つの中の整数ゼロ（0）は、制御域500の対応ポインタによって指示されるバッファが割当て済みであることを示す。そのフィールドの整数1は、バッファが割当てられていないということ指示する。ステップ612では、一時的変数TEMPが0に等しくセットされる。次に、ステップ613で、A[index]中の割当てフラグがTEMPとアトミック・スワップされ、割当てフラグがクリアされる。本発明の実施の形態においては、SPARCブランドのマイクロプロセッサ上で利用可能なような「アトミック・スワップ」動作を使用する。この動作は、原子的に、すなわち途中の割込み、据置きトラップ、またはシステム内の他のスレッドが割当てフラグA[index]にアクセスするのを許容することなく行われる。このように、この領域にアクセスする他のプロセスは、値がスワップされ終わるまで、ロックアウトされる。アトミック・スワップについては、例えば、米国カリフォルニア州メンロパーク（Menlo Park）のSPARC International社から入手可能なSPARCアーキテクチャ・マニュアル（Architecture Manual）（バージョン8、1992年刊）の102～103ページに記載されている。

【0027】次に、ステップ614では、alloc/dealloc（あるいはレジスタ）から取り出された値（ステップ614ではTEMP変数）が1に等しいかどうか判断される。この値が1に等しい場合は、バッファが割当てに利用可能であり、ステップ618で、バッファのポインタが要求プロセスにリターンされる。1に等しくない場合は、ステップ616で、カウンタが1だけディクリメントされ、次のalloc/deallocを調べるためにインデックスが2だけインクリメントされる。ステップ608～616は、制御域によって参照された各バッファについてalloc/deallocフラグをチェックすることによって、利用可能なバッファが検出されるまで続けられる。このように、2つのプロセス（例えば、クライアント/カーネルまたはカーネル/サーバ）間の通信のために指定された通信領域

からのバッファの割当てを容易に行うことができる。このプロセスは、図4によって上に説明した割当てステップにとって特に役に立つ。

【0028】次に、バッファの割当て解除について図7のプロセス700を参照して説明する。割当て解除プロセスは、ステップ702〜708では前述のプロセス600と同様に進行し、まず最初にバッファのポインタbを受け取って、割当て済みバッファの数A[0]を取り出し、ステップ702で、インデックスを1に初期化する。次に、ステップ704で、カウンタの内容が範囲内にあるかどうかチェックされ、範囲内になれば、ステップ706で、エラーを伴うプロセスからのリターンが発生する。ステップ708では、制御域によって参照された中で割当て済みでないバッファがあるかどうか判断される。割当て済みでないバッファがなければ、すなわちカウンタの内容が0に等しければ、ステップ710でエラーがリターンされる。ステップ712では、制御構造における参照により指示されたバッファが、割当て解除しようとするバッファのポインタbに等しいかどうか判断される。これが等しければ、対応する割当てフラグA[index]が1に等しくセットされ、バッファが現在割当て解除されていて、他のプロセスのために試用可能であるということを指示する。次に、プロセスは、ステップ718で、動作が首尾よく終了したことを示すリターン引数（例えばOK）と共にリターンする。上記の対応するポインタが、ステップ712での判断に基づき、割当て解除しようとするバッファを指示しない場合は、ステップ714において、インデックスが2だけインクリメントされ、カウンタが1だけデクリメントされる。ステップ708〜714は、割当て解除しようとするバッファがステップ712で決定されるか、ステップ708でチェックしようとするバッファがもはや残っていない（カウンタ=0）になるまで繰り返される。

【0029】次に、最後のプロセスとして、クライアントとカーネルの間、あるいはカーネルとサーバ・プロセスの間の通信のための共用メモリエリアの初期化のような、本発明の一実施の形態における2つのプロセス間の共用メモリエリア（図8のプロセス・フローチャート800ではメモリエリアAと称する）の初期化について説明する。この場合も、前述の場合と同様に、nのバッファ全部の割当てをクライアント・プロセスに入ると同時にすることもできれば、各バッファに必要な応じて個別にメモリを割当てすることも可能である。図8に示すように、ステップ802においては、制御域500の第1の要素（例えば、501）における利用可能なバッファの数A[0]がnに等しくセットされる。実施の形態においてはn=32であるが、設計上の選択事項に従って任意の数のバッファを使用することが可能である。次に、対応するカウンタもnに等しくセットされ、インデック

ス変数が1に等しくセットされる。次に、ステップ804で、nの全てのバッファについてメモリがオペレーティング・システムから割当てられたかどうか判断される。その結果がノーの場合、プロセス800はステップ808に進む。ステップ808では、対応するバッファのための割当てフラグが1に等しくセットされ、そのバッファが使用可能であることを指示する。さらに、バッファAの対応参照[index]+1が、一部のオペレーティング・システムでallocate_buffer()と命名されているようなメモリ割当て基本要素に等しくセットされる。この例においては、関数allocate_bufferは、例えば5キロバイト長のメモリ領域を割当てることができるが、バッファのサイズは、設計上の選択事項により任意の大きさとすることが可能である。

【0030】ステップ808においてバッファ使用可能であるという指示が出、そのバッファに適切なメモリが割当てられたならば、インデックスが2だけインクリメントされ、カウンタは1だけデクリメントされる。プロセス・ステップ804〜810は、クライアントとカーネルまたはカーネルとサーバのような2つのプロセス間の共用メモリ領域で総数nのバッファが適切なスペースを割当てられたことが検出されるまで繰り返される（割当てられるバッファの数がnに等しい場合）。ステップ804でカウンタの内容が0に等しいことが検出された場合は、プロセスは終了し、ステップ806にリターンする。

【0031】このように、本発明による上記の技術を用いることによって、2つのプロセスの間の通信のための制御域とバッファを生成し、プロセス間通信に用いることができる。本発明は、所与の任意の時点でいくつかのプロセス・スレッドがアクティブになり得るような状況において特に有用であり、メモリの使用を従来技術におけるよりも数段効率的にすることができる。以上、本発明に関する特定の実施の形態を特に図面を参照しつつ説明したが、当業者には、本発明の要旨及び範囲から逸脱することなく本発明の変更態様や修正態様を達成することが可能なことは明白であろう。従って、本発明は特許請求の範囲の記載によってのみ限定されるものである。

【図面の簡単な説明】

【図1】 従来技術によるプロセス間通信の方法を示すブロック図である。

【図2】 本発明の実施の形態を実装することができるコンピュータシステムを示すブロック図である。

【図3】 コンピュータシステムにおけるプロセス及びそれらの各プロセスに割当てられるバッファを示すブロック図である。

【図4】 本発明の実施の形態におけるクライアント・プロセス、カーネル及びサーバ・プロセス中の一連のステップを示す説明図である。

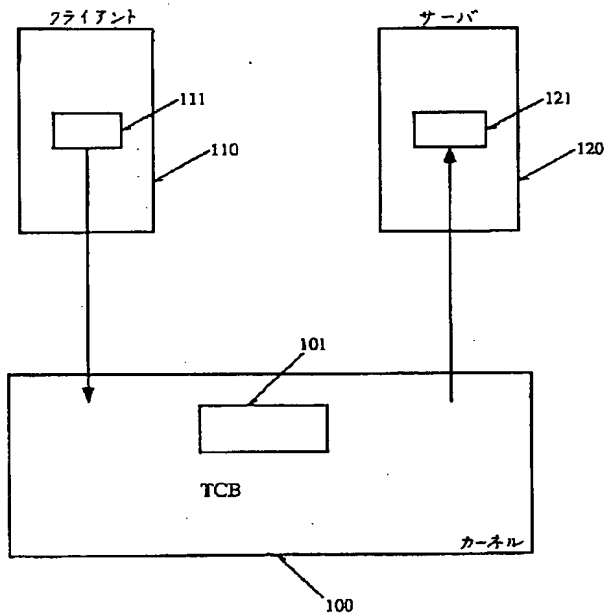
17

【図 5】 プロセス間通信に使用されるバッファの詳細な構造を示す説明図である。

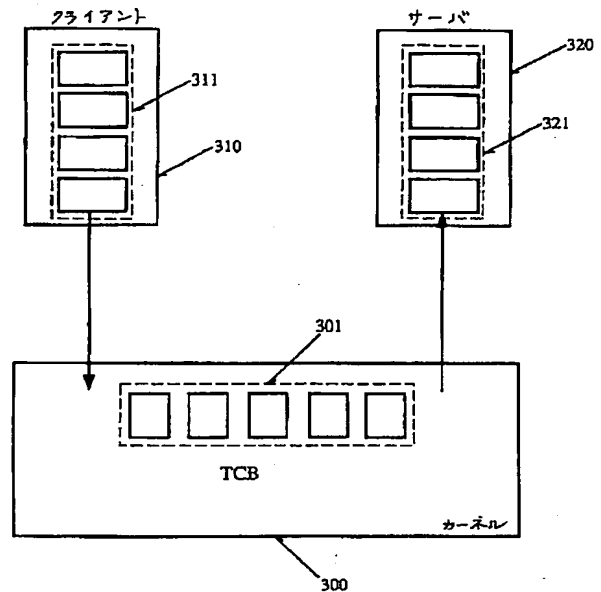
【図 6】 図 6 は、2つのプロセス間で通信するために用いられる指定されたエリアの中でバッファを割当ててする方法を示すフローチャートである。

【図 7】 2つのプロセス間で通信するための指定されたエリアの中でバッファを割当て解除するための方法を示すフローチャートである。

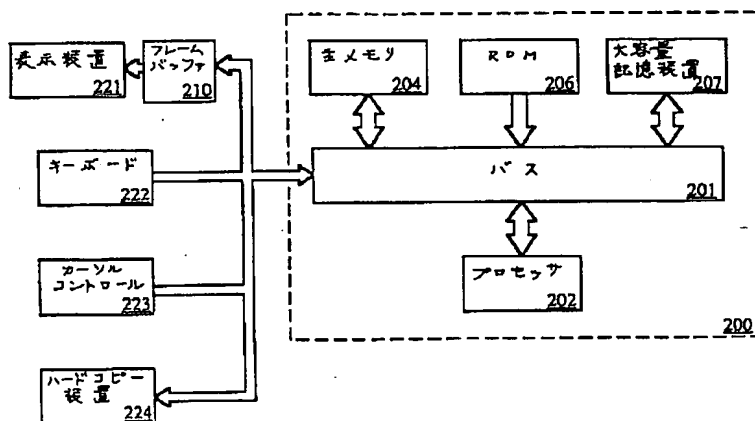
【図 1】



【図 3】



【図 2】



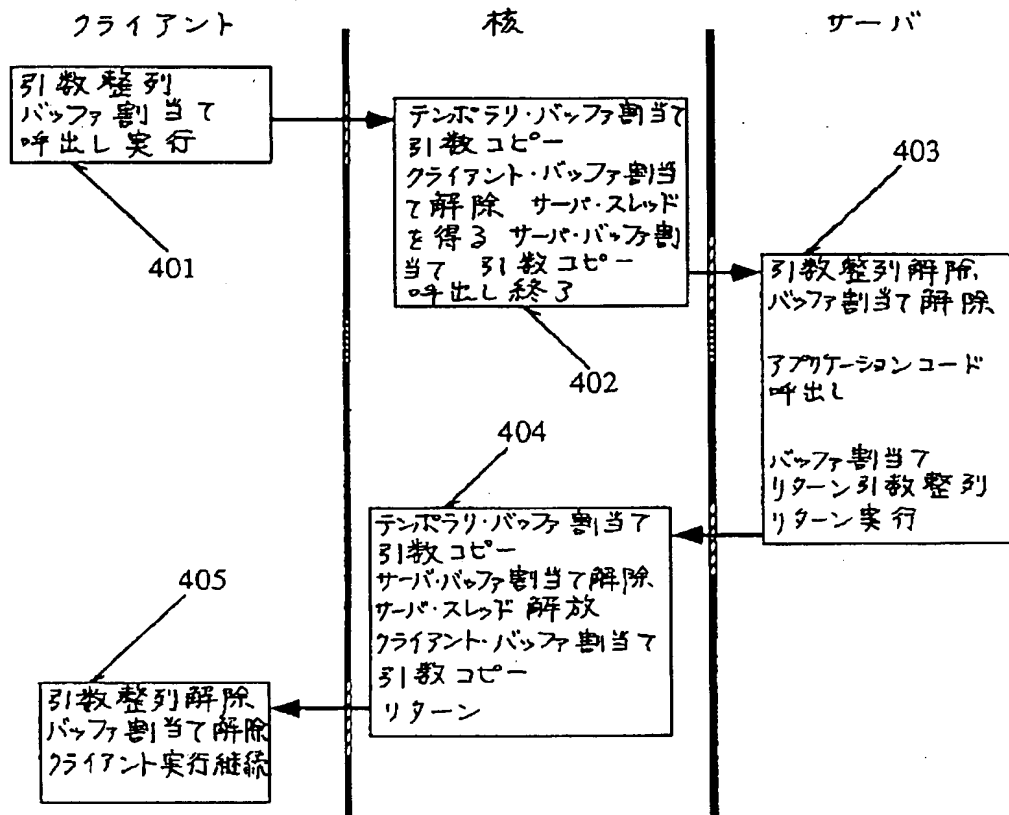
18

【図 8】 メモリをプロセス間通信に割当ててする方法を示すフローチャートである。

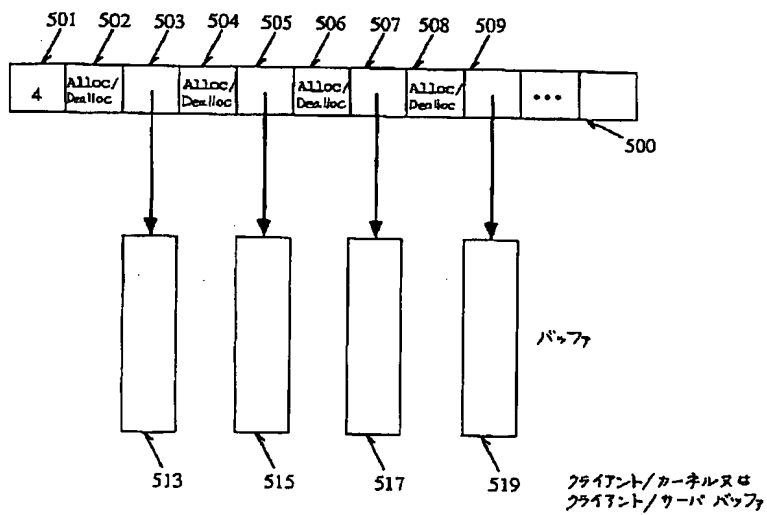
【符号の説明】

200 コンピュータシステム、201 バス、204 主メモリ、206 ROM、207 大容量記憶装置、210 フレームバッファ、221 表示装置、222 キーボード、223 カーソル・コントロール、224 ハードコピー装置。

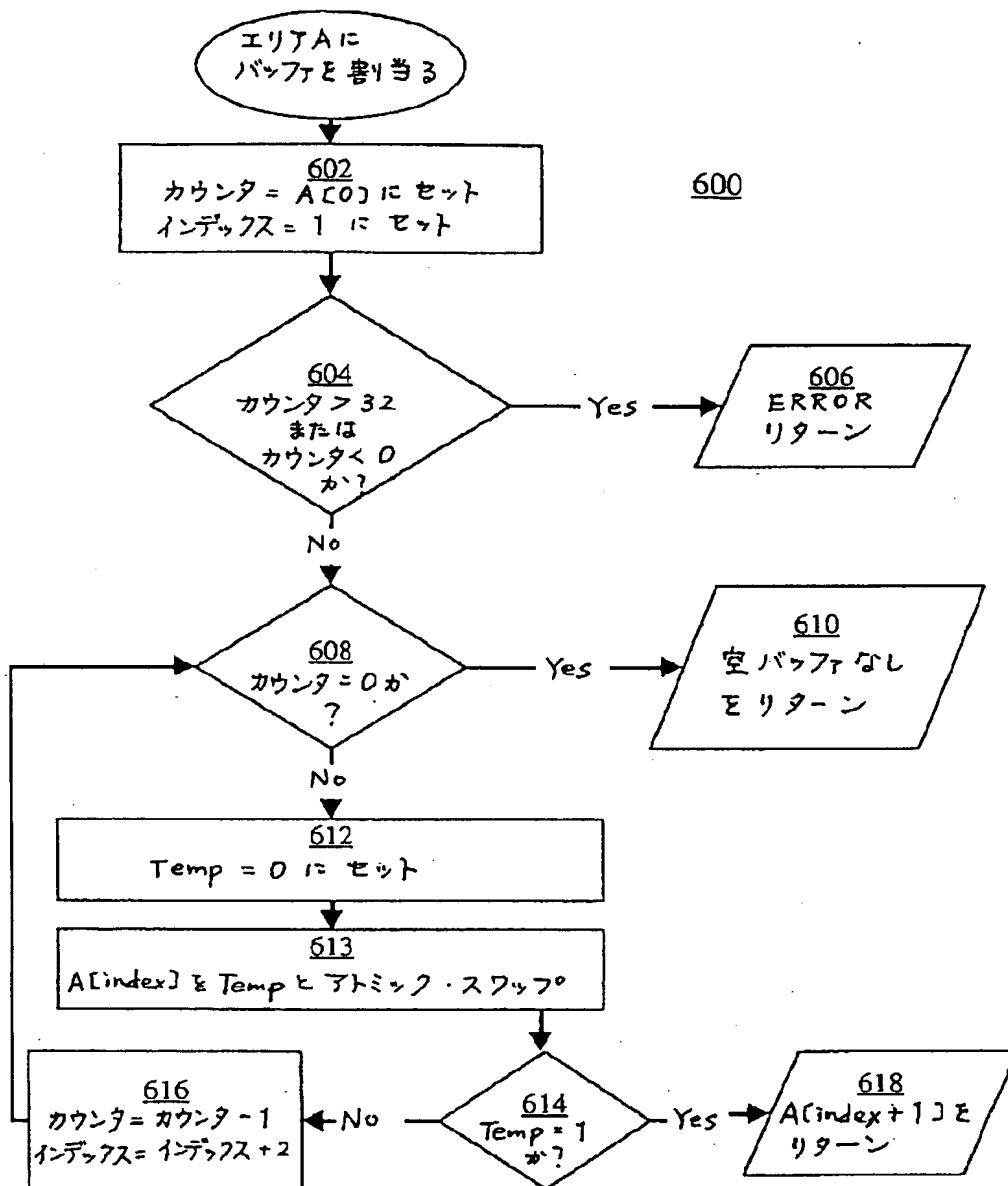
【図 4】



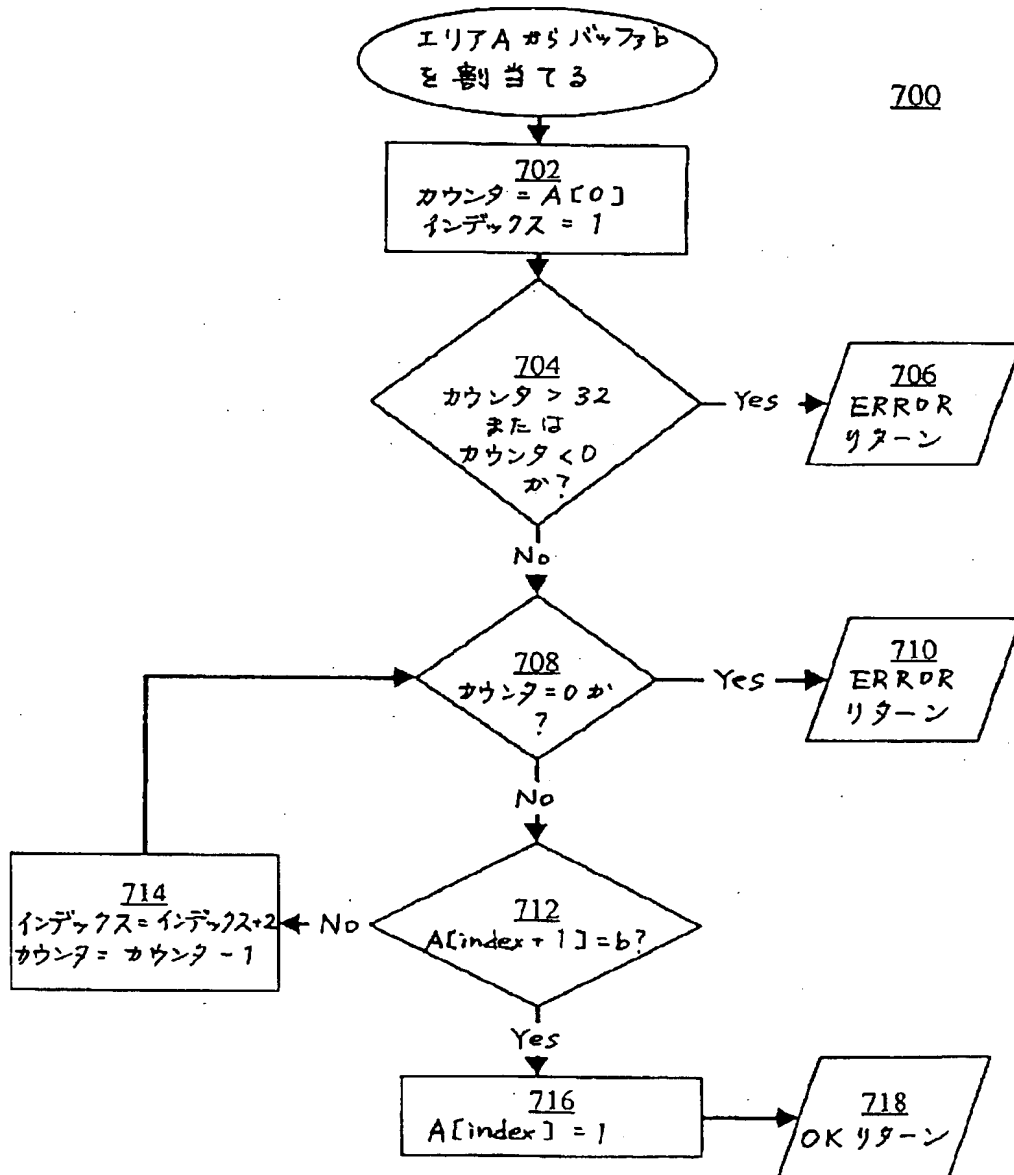
【図 5】



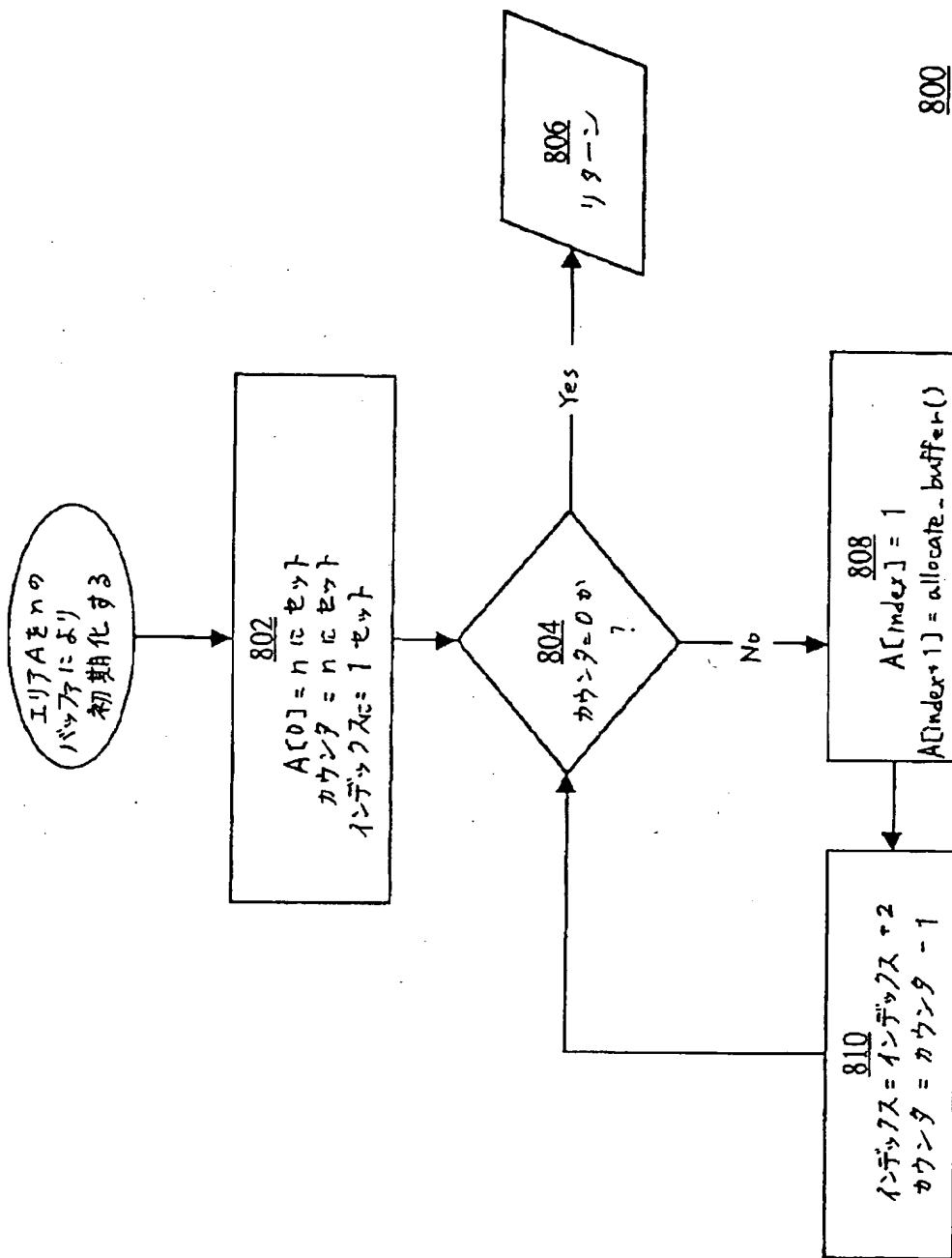
【図6】



【図7】



【図 8】



フロントページの続き

(72) 発明者 グラハム・ハミルトン
 アメリカ合衆国 94303 カリフォルニア
 州・パロ アルト・デビッド コート・
 3143